- void setRest(Structure tail)
- void setFirst(Structure dat)
- Object removeFront()
- void insertFront(Structure dat)

*LRStructure supports additional operations on the list structure:

**LRStructure: An Enhanced List**
\textbf{Constructing LRStruct from Allist}
... System.out.println(altL);

List list = ListFactory.Singleton.makeEmptyList();
Alt list = list;
Alt list = list;
Alt list = list;

... }
...  

```java
System.out.println(att);  
I1st.insertFront(new Integer(-1));  
LRstruct att = I1st;  
LRstruct I1st = new LRstruct();  // an empty list
...  
```

• What is printed?

Contrasting LRstruct from ALI1st (cont.)
- This instance represents the current state of the system.

Represent the system by a class containing an instance of a concrete state.

- Each concrete state must implement its own concrete methods.

Define a concrete subclass of the abstract class for each state of the system.

Define an abstract class for the states of the system.

The State Pattern
If it can change its class dynamically:

- Since this instance can change dynamically, the system will behave as

  • Delegate all requests made to the system to the current state instance.

  • Define methods for the system to return the current state and to change

  state.

The State Pattern (cont.)